




## Ejercicio Práctico

 **Título:** *Implementación Básica de Autenticación y Autorización en una Aplicación Web*

---

### **Objetivo del ejercicio:**

Crear un sistema sencillo de **autenticación** y **autorización** para una aplicación web que permita a los usuarios registrarse, iniciar sesión con una contraseña y acceder a recursos protegidos basados en su rol.

---

### **Escenario:**

Tienes que desarrollar una aplicación web simple para un sistema de **gestión de tareas**. El sistema tendrá dos tipos de usuarios:

- **Usuario Común:** Puede ver y agregar tareas, pero no puede editar ni eliminar tareas.
- **Administrador:** Puede ver, agregar, editar y eliminar tareas.

El sistema debe permitir a los usuarios autenticarse con su nombre de usuario y contraseña, y restringir el acceso a las funciones según su rol.

---

### **Tu tarea:**

#### **Paso 1 – Crear la página de registro:**

1. **Formulario de registro:** El formulario debe permitir a los usuarios ingresar su nombre de usuario y contraseña.

- Al registrarse, el sistema debe almacenar las credenciales de forma segura utilizando **hashing** (puedes usar **bcrypt** para el hash de contraseñas).
- Los usuarios deben ser asignados al **rol de Usuario Común** por defecto.

## Paso 2 – Crear la página de inicio de sesión:

1. **Formulario de inicio de sesión:** Los usuarios deben ingresar su nombre de usuario y contraseña.
  - El sistema debe verificar que las credenciales coincidan con las almacenadas en la base de datos.
  - Al iniciar sesión, el sistema debe crear una **sesión** para el usuario (puedes usar **sesiones de servidor** o **JWT** para este ejercicio básico).

## Paso 3 – Crear las funciones de tareas:

1. **Ver tareas:** Ambos tipos de usuarios deben poder ver las tareas que han sido almacenadas en el sistema.
2. **Agregar tareas:** Los **Usuarios Comunes** y los **Administradores** deben poder agregar tareas. Sin embargo, los **Administradores** tienen permisos para **editar** y **eliminar** tareas.

## Paso 4 – Control de acceso según roles:

1. **Verificación de roles:**
  - Si el usuario es un **Administrador**, debe tener acceso completo (ver, agregar, editar y eliminar tareas).
  - Si el usuario es un **Usuario Común**, debe poder ver y agregar tareas, pero no editar ni eliminar ninguna.

## Paso 5 – Cierre de sesión:

1. Implementa una **función de cierre de sesión** que destruya la sesión activa del usuario y lo redirija a la página de inicio de sesión.

---

 **Resultado esperado:**

- Un sistema básico que permita a los usuarios **registrarse e iniciar sesión** de manera segura.
  - Un sistema de **roles** que permita a los **Administradores y Usuarios Comunes** acceder a diferentes funcionalidades (ver, agregar, editar, eliminar tareas).
  - **Control de acceso** que restrinja a los **Usuarios Comunes** de realizar acciones no autorizadas.
  - Función de **cierre de sesión** que elimine la sesión y redirija a la página de inicio de sesión.
- 



### Entrega sugerida:

1. **Código fuente** de la implementación de la autenticación y autorización.
  2. **Capturas de pantalla** mostrando el flujo de registro, inicio de sesión y funciones de tareas con control de acceso.
  3. **Informe breve** que explique cómo se implementó la gestión de sesiones y el control de roles.
- 



### Herramientas recomendadas:

- **Node.js y Express.js** para el backend (puedes usar cualquier otro framework backend de tu preferencia).
  - **bcrypt** para el hashing de contraseñas.
  - **MongoDB o MySQL** para almacenar usuarios, contraseñas y tareas.
  - **Postman o Insomnia** para probar las rutas de la API.
- 



### Sugerencia para extender el ejercicio (opcional):

- **Autenticación multifactor (MFA)**: Implementa un sistema de autenticación más seguro utilizando un segundo factor (como un código enviado por correo electrónico o SMS).

- **Validación y filtrado de entradas:** Implementa medidas de seguridad como la **validación** y el **filtrado** de entradas para evitar **inyección SQL** y **XSS**.
-